12-2011

Parallel machines scheduling with applications to Internet ad-slot placement

Shaista Lubna
University of Nevada, Las Vegas

PARALLEL MACHINES SCHEDULING WITH APPLICATIONS TO INTERNET

AD-SLOT PLACEMENT

By

Shaista Lubna

Bachelor of Engineering in Computer Science and Information
Technology
JNTU University, India
May 2006

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree in Computer Science

School of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
December 2011

THE GRADUATE COLLEGE

We recommend the thesis prepared under our supervision by

Shaista Lubna

entitled

Parallel Machines Scheduling with Applications to Internet Ad-Slot Placement

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
School of Computer Science

Wolfgang Bein, Committee Chair

Ajoy Datta, Committee Member

Lawrence Larmore, Committee Member

Emma Regentova, Graduate College Representative

Ronald Smith, Ph. D., Vice President for Research and Graduate Studies
and Dean of the Graduate College

December 2011

ABSTRACT

Parallel machines scheduling with applications to Internet ad-slot placement.

by
Shaista Lubna

Dr. Wolfgang Bein, Examination Committee Chair
Professor, Department of Computer Science
University of Nevada, Las Vegas

We consider a class of problems of scheduling independent jobs on identical, uniform and unrelated parallel machines with an objective of achieving an optimal schedule. The primary focus is on the minimization of the maximum completion time of the jobs, commonly referred to as Makespan (C_{max}). We survey and present examples of uniform machines and its applications to the single slot and multiple slots based on bids and budgets.

The Internet is an important advertising medium attracting large number of advertisers and users. When a user searches for a query, a search engine returns a set of results with the advertisements either on top of the page or on the right hand side. The assignment of these ads to positions is determined by an auction using the ad-slot placement. The algorithmic approach using the level algorithm (which constructs optimal preemptive schedules on uniform parallel machines) is taken into consideration for assigning bidders to the slots on the Internet.

ACKNOWLEDGEMENTS

I would like to thank Dr. Wolfgang Bein for chairing my committee and advising this work. I am thankful for his continuous guidance and help to deepen my work. Without his generous help this thesis would not have had such a rich content. I am thankful to Dr. Ajoy K Datta for his moral support and guidance through my Masters program and help on my thesis. I would also like to specifically thank Dr. Lawrence Larmore and Dr. Emma Regentova for serving on the committee. For this and for being generous with their time when I needed it, I am deeply indebted to them. I would like to thank the faculty at the School of Computer Science, University of Nevada, Las Vegas for the formal education along with generous financial support.

I would also like to extend my appreciation towards my family for being there for me through thick and thin and always encouraging me to strive for the best. Without their endless support I would never be able to reach to the place I’m standing today in my life. Last but not the least; I thank my friends, for their support in the successful completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In most manufacturing systems, a decision-making process that plays a crucial role consists in allocating the time at which a particular task is to be processed by a given resource in order to optimize the requirements set by the customer. This function is referred to as *scheduling*. Indeed, the current economic and commercial market pressures (the growing consumer demand for variety, reduced product life cycle, changing markets with global competition, rapid development of new processes and technologies, etc...) emphasize the need for a system which requires only small inventory levels, minimizes waste production but is able to maintain customer satisfaction by delivering the required goods at the specified time. This requires efficient, effective and accurate scheduling, which is a complex operation in almost all production environments. The importance of scheduling is exemplified by an investigation carried out in the United States mechanical industrial sector which shows that the machines spend about 80% of their total processing time in waiting for the tasks.

Scheduling theory is generally concerned with the optimal allocation of scarce resources to activities over time. More formally, scheduling problems involve jobs that must be scheduled on machines subject to certain constraints to optimize some objective function. A schedule is for each job an allocation of one or more time intervals to one or more machines [2]. Schedules may be represented by **Gantt charts** as shown in Figure 1.

A Gantt chart is a type of bar chart that illustrates a project schedule and may be machine oriented or job oriented [2]. (a) and (b) denote the Machine and job oriented Gantt charts respectively.

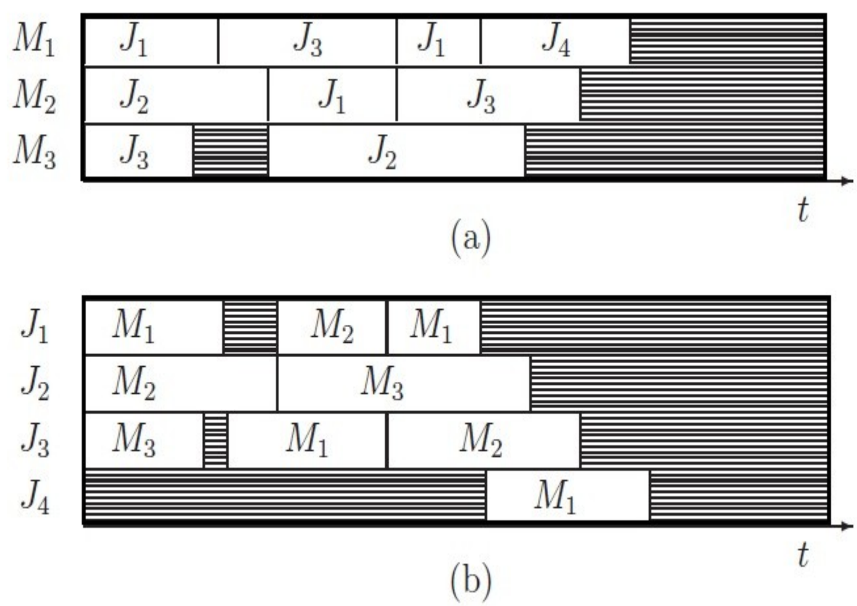

Figure 1 : Machine and job-oriented Gantt charts.

Graham et al. (1979) introduced the standard α | β | γ notation for representing scheduling problems. This notation embodies the three main elements which define the scheduling problem: the machine environment, the job characteristics, and the optimization criterion. In the sequel, we briefly detail these three fields. In the considered scheduling models, the number of machines and the number of jobs are assumed to be finite and fixed.

There are several machine environments (represented by the field α) which are summarized in the following:

• Single machine ($\alpha = 1$): The process of assigning various jobs to one machine.

• Parallel machines (α = P or Q or R): Each job requires a single operation to be performed on one out of a set of available machines.

• Flow shop (α = F): There are several machines in series. Each job has to be processed on each one of the machines. All jobs have the same routing.

• Job shop (α = J): This model is similar to the flow shop, with the only difference that each job has its own route to follow.

• Open shop (α = O): Likewise the job shop, each job has to be processed on each one of the machines. However, there is no restriction on the routing of each job. The scheduler is allowed to determine the route of any job [1].

Several possible job characteristics (represented by the field β) may modify the scheduling environment. Some of these characteristics are:

• Preemption (pmtn): The processing of any operation may be interrupted and resumed at a later time.

• Precedence constraints (prec): A precedence relation between jobs requires that one or more jobs have to be completed before another job is allowed to start its processing.

• Release dates or heads (r_j): No job can start its processing before its release date.

• Delivery times or tails (q_j): After finishing its processing, each job has to spend an amount of time before exiting the system [1].

The goal of a scheduling algorithm is to produce a ”good” schedule, but the definition of ”good” will vary depending on the application. Therefore, an optimization criterion (represented by the field γ) has to be specified. The most commonly chosen criteria involve the minimization of:

• Makespan (C_{max}): The completion time of the last job to leave the system.

• Maximum lateness (L_{max}): The worst violation of the due dates. The job lateness is non-negative if it is completed late and negative otherwise.

• Maximum tardiness (T_{max}): The difference between tardiness and lateness is that tardiness equals zero if the job is completed early (i.e. $T_{max} = \max(0, L_{max})$).

• Maximum flow time (F_{max}): The flow time of a job denotes the time elapsed between its entry to its exit from the system.

• Total (weighted) completion time (ΣC_j or $\Sigma w_j C_j$):The sum of the (weighted) completion times. It indicates the total holding (or inventory) costs incurred by the schedule. This criterion is equivalent to the total (weighted) flow time criterion.

• Total (weighted) tardiness (ΣT_j or $\Sigma w_j T_j$): It is a more general cost function than the total (weighted) completion time .

• (Weighted) Number of tardy jobs (ΣU_j or $\Sigma w_j U_j$): A job is considered as tardy if it is completed after its due date [1].

Thesis Overview: In chapter 2 we survey the types of Parallel machines and approximation algorithms. The applications of the level algorithm is presented in detail in Chapter 3, with suggestive examples. Ad-slot mechanism is reviewed in Chapter 4 with single slot and multiple slots and its illustration. We finish with concluding remarks in Chapter 5.

CHAPTER 2

PARALLEL MACHINES

Given a set of n jobs $J_i(i=1,\ldots,n)$ to be processed on m parallel machines $M_j(j=1,\ldots,m)$. Each job J_i has a processing requirement $P_i(i=1,\ldots,n)$ and every machine has a speed $S_j(j=1,\ldots,m)$. Each job requires a single operation to be performed on one out of a set of available machines. The goal is to attain an optimal schedule that specifies when and on which machine each job is to be executed.

The following examples illustrate the role of parallel machines in two different real-life situations.

Example 2.1: Consider the central processing unit of a computer that must process a sequence of programs (jobs) that arrive over time. In what ordering should the programs be processed in order to minimize the average completion time?

Example 2.2: Consider a factory that produces paper bags for cement, charcoal, dog food, and so on. The basic raw material for such an operation is rolls of papers. The production process consists of three stages: printing the logo, gluing the side of the bag, and sewing up one end or both ends. The different bags require different amounts of processing times on different machines. The factory has orders for batches of bags; each order has a date by which it must be completed.

In what ordering should the machines work on different bags in order to ensure that the factory completes as many orders as possible on time?

Parallel Machines can be divided into three classes:

•*Identical parallel machines (α = P)*: All the available machines have the same speed.

•*Uniform parallel machines (α = Q)*: The machines have different speeds, but these speeds are independent of the jobs.

•*Unrelated parallel machines (α = R)*: The machines have different speeds, but these speeds are dependent of the jobs [1].

2.1 Identical Parallel Machines :

We consider the problem of scheduling independent jobs on identical parallel machines. Formally there are *n* jobs $J_i (i=1,\ldots,n)$ with processing times $p_i (i=1,\ldots,n)$ to be processed on *m* identical parallel machines $M_1,\ldots,M_m$ [2].

$m = 3$

i	1	2	3	4	5
p_i	4	5	3	5	4

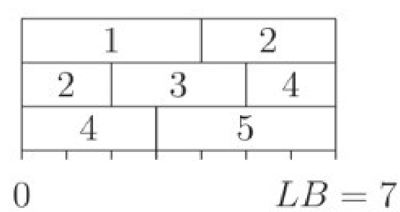

Figure 2 : Optimal schedule for an instance of P | pmtn | C_{max} .

Mc Naughtons wrap around rule : Compute $D = max\{max\ p_i, (1/m) p_i\}$. Assign the jobs in any order from time 0 until time D on machine. If a jobs processing extends beyond time D, preempt the job at time D, and continue its processing on next machine, starting at time 0. Repeat this process until all jobs are assigned [7][18].

2.1.1 P | pmtn | C_{max}

Theorem 1: Mc Naughtons wrap around rule is optimal for P | pmtn | C_{max} [7].

Proof: It is clear that D is a lower bound for the optimal schedule length. If we can show that wrap around rule can always generate a feasible schedule in the time interval [0,D],then the schedule must be optimal.

i) $D \geq max\{P_i\}$ no jobs can overlap i.e.; simultaneously execute on more than one machine.

ii) $mD \geq \{P_j\}$ as there is enough capacity in the time interval [0,D] to schedule all jobs.

Thus a wrap around rule can always generate a feasible schedule can be constructed in O(n) time.

2.1.2 P | pmtn; r_i | L_{max}

Each job J_i has a release time r_i and a due date d_i with $r_i \leq d_i$. To find a preemptive schedule on m identical machines such that the maximum lateness L_i is defined as $max_{i=1}^{n}\{C_i - d_i\}$ is minimized.

Taking in to account the decision version of the problem: Given some threshold value L there exist a schedule such that

$$max_{i=1}^{n} L_i = max_{i=1}^{n}\{C_i - d_i\} \leqslant L \qquad (1)$$

The above relation holds if and only if

$$C_i \leqslant d_i^L := L + d_i \; for \; all \; i = 1, \ldots, n.$$

All jobs i must finish before the modified due dates d_i^L and cannot start before the release times r_i , i.e. each job J_i must be processed in an interval $[r_i, d_i^L]$ associated with J_i . These intervals are called **time windows** [2]. We approach the general problem of finding a preemptive schedule for jobs $J_i(i=1,\ldots,n)$ on m identical machines such that all jobs J_i are processed within their interval or time windows $[r_i, d_i]$ by reducing to a maximum flow problem in a network constructed as follows.

Let

$$t_1 < t_2 < \ldots < t_r$$

be the ordered sequence of all different r_i values and d_i values. Consider the intervals

$$I_K := [t_K, t_{K+1}] \text{ of length } T_K = t_{K+1} - t_K \text{ for } K = 1, \ldots, r-1.$$

We associate a job vertex with each job J_i and an interval vertex with each interval. In addition to the existing nodes we add two dummy vertices source node 's' and target node 't'. Between these vertices, arcs and capacities for these arcs are defined as follows. From s we have an arc to each job vertex J_i with capacity p_i and from each interval vertex I_K we have an arc to t with capacity mT_K .

There exists an arc from J_i to I_K if job J_i can be processed in I_K , i.e. iff $r_i < t_K$ and $t_{K+1} < d_i$. The capacity of this arc is T_K . It is not difficult to prove that there exists a schedule respecting all time windows if and only if the maximum flow in N has the value $\Sigma_{i=1}^{n} p_i$. If this is the case, the flow x_i on the arc (J_i, I_K) corresponds with the time period in which job J_i is processed in the time interval I_K and we have

$$\Sigma_{K=1}^{r-1} x_{i_K} = p_i \text{ for } i = 1, \ldots, n. \quad (2)$$

$$\Sigma_{i=1}^{n} x_{i_K} \le mT_K \text{ for } K = 1, \ldots, r-1. \quad (3)$$

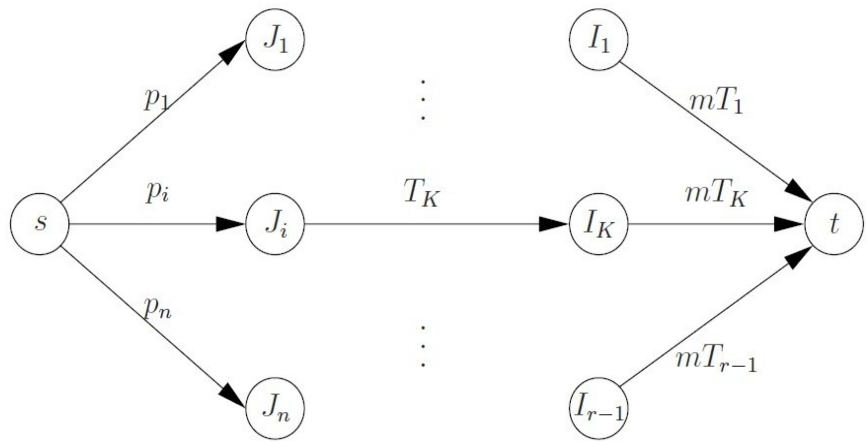

Figure 3: A network for problem $P \mid pmtn; r_i \mid L_{\max}$.

Therefore each job is completely processed and the total amount of processing time in I_K is at the most mT_K, which is the capacity of m machines.

Furthermore, $x_{iK} \leq T_K$ for all (J_i , I_K) ∈ A. (4)

Then there exists a maximal flow satisfying, a feasible solution for the scheduling problem with time windows is constructed by scheduling partial jobs J_{iK} with processing times $x_{iK} > 0$ in the intervals I_K on m identical machines.

For each K, this is essentially a P | pmtn | C_{max} problem, which has a solution with $C_{max} \leq T_K$ because of (3) and (4).

Because network N has at the most O(n) vertices, the maximum flow problem can be solved in $O(n^3)$ time. Furthermore, the schedule respecting the windows can be constructed in $O(n^2)$ time. Thus, the window problem can be solved in $O(n^3)$ steps [2].

Example: Consider the problem $P \mid r_i \mid L_{max}$ on three machines. Given are processing times $p_1 = 2$, $p_2 = 2$, $p_3 = 3$, $p_4 = 2$. $r_1 = 0$, $r_2 = 1$, $r_3 = 4$, $r_4 = 1$. $d_1 = 5$, $d_2 = 8$, $d_3 = 6$, $d_4 = 8$. and let the threshold value L be 3. Use the network flow method with time windows to see if there exists a feasible schedule for the problem L=3. If yes, Draw the schedule.

Solution:

(i) Modify the due dates by $d_L = L + d_i$. We have

$d_1 = 5 + 3 = 8$.

$d_2 = 8 + 3 = 11$.

$d_3 = 6 + 3 = 9$.

$d_4 = 8 + 3 = 11$.

(ii) Unions of Release times and due dates are 0, 1, 4, 8, 9, 11.

The time windows derived are [0,1] [1,4] [4,8] [8,9] [9,11].

$I_K := [t_K , t_{K+1}]$ of length $T_K = t_{K+1} - t_K$ for K = 1, . . . r. There exists an arc between J_i and I_k iff job J_i can be processed in I_k i.e; iff $r_i \le T_k$ and $T_{k+1} \le d_i$. The capacity is T_k.

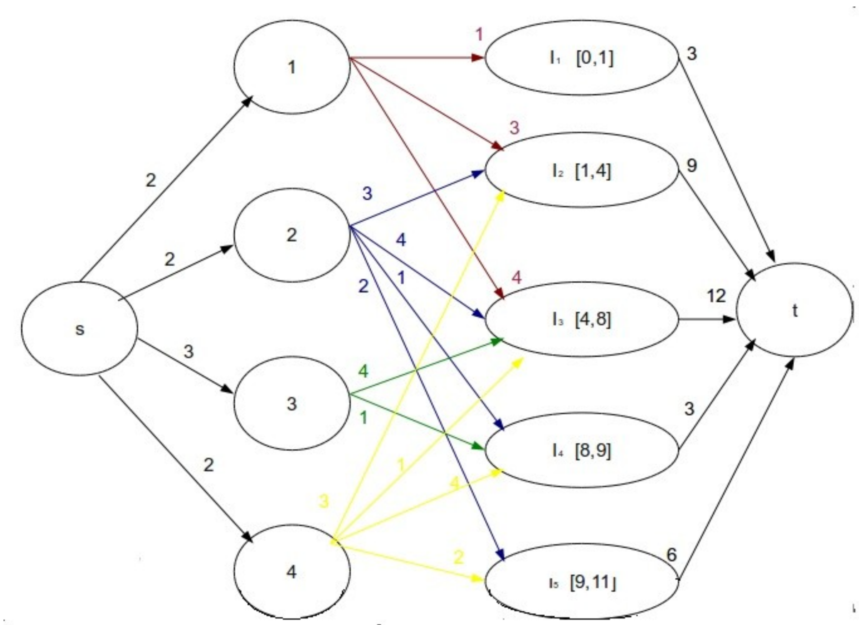

Figure 4 : Example of network for problem P | pmtn r_i | L_{max}

flow [J_1 , I_1] = 1

flow [J_1 , I_2] = 1

flow [J_2 , I_2] = 2

flow [J_3 , I_3] = 3

flow [J_4 , I_3] = 2

ΣP_i = 2 + 2 + 3 + 2 = 9

The Optimal Schedule is

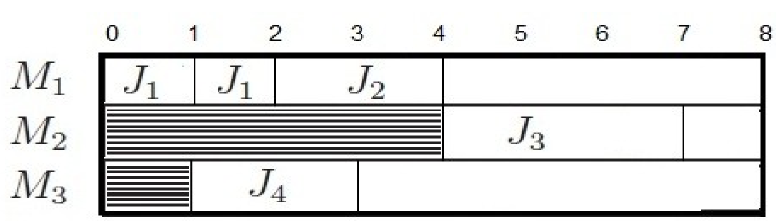

Figure 5 : Schedule for problem P | pmtn r_i | L_{max}

2.2 Unrelated Parallel Machines

We have n independent jobs i = 1, . . . , n to be processed on m machines. The processing time of job i on machine M_j is p_{ij} (i = 1, . . . , n; j = 1, . . . , m). This model is a generalization of the uniform machine model we get by setting $p_{ij} = p_i / s_j$ which is explained in the next section.

2.2.1 R || ΣC_i

R || ΣC_i is reduced to an assignment problem[2]. If i_1 , i_2 , . . . , i_r is the sequence of jobs processed at machine M_j, then the contribution of these jobs in the objective function is

$$r\, p_{i_1 j} + (r-1)\, p_{i_2 j} + \ldots + 1\, p_{i_r j}$$

We define a position of a job on a machine by considering the job processed last on the first position, the job processed second from last on the second position, etc. To solve problem R || ΣC_i we have to assign the jobs i to positions k on machines j. The cost of assigning job i to (k, j) is kp_{ij} . Note that an optimal solution of this assignment problem has the following property: if some job i is assigned to position k > 1 on machine j, then there is also a job assigned to position k − 1 on machine j. Otherwise, scheduling job i in position k − 1 would improve the total assignment cost (provided that $p_{ij} > 0$). Thus, solution of the assignment problem always yields an optimal solution of our scheduling problem.

2.2.2 $R \mid pmtn \mid C_{max}$, $R \mid pmtn \mid L_{max}$ and $R \mid pmtn; r_i \mid L_{max}$

We solve problem $R \mid pmtn \mid C_{max}$ in two steps. In the first step we formulate a linear program to calculate for each job i and each machine j the amount of time t_{ij} machine j works on job i in an optimal schedule. In a second step, a corresponding schedule is constructed. First we give the linear programming formulation. Problem $R \mid pmtn \mid C_{max}$ is given by nm positive integers p_{ij} , which represents the total processing time of job i on machine M_j. Let t_{ij} be the processing time of that part of job i which is processed on M_j. Then t_{ij} / p_{ij} is the fraction of time that job i spends on machine j, and the equation

$$\Sigma_{j=1}^{m} \frac{t_{ij}}{p_{ij}} = 1$$

must hold in order for job i to be completed (i = 1, . . . , n).

This leads to the following formulation of the problem:

minimize C_{max}

subject to

$$\Sigma_{j=1}^{m} \frac{t_{ij}}{p_{ij}} = 1, \quad i = 1 \ldots n. \quad \text{(a)}$$

$$\Sigma_{j=1}^{m} t_{ij} \le C_{max} \quad i = 1 \ldots n. \quad \text{(b)}$$

$$\Sigma_{i=1}^{n} t_{ij} \le C_{max} \quad j = 1 \ldots m. \quad \text{(c)}$$

$$t_{ij} \geq 0 \;\; i = 1 \ldots n;\; j = 1 \ldots m.$$

The left-hand side of (b) represents the time job i (i = 1, . . . , n) spends on all machines. The left-hand side of (c) represents the total time machine M_j (j = 1, . . . , m) spends processing jobs. Note that for an optimal solution of this linear program we have

$$C_{max} = max\{max_{i=1}^{n} \Sigma_{j=1}^{m} t_{ij}, max_{j=1}^{m} \Sigma_{i=1}^{n} t_{ij}$$

The problem of finding a corresponding schedule is equivalent to the problem of finding a solution to the preemptive open shop problem with processing times t_{ij} (i = 1, . . . , n; j = 1, . . . , m) which has a C_{max} value given by (4). We conclude that problem R | pmtn | C_{max} is polynomially solvable.

A similar approach may be used to solve R | pmtn | L_{max}. We formulate a linear programming problem to minimize L_{max}.

Assume that the jobs are numbered in nondecreasing due date order, i.e. $d_1 \leq d_2 \leq . . . \leq d_n$.

Let $t_{ij}^{(1)}$ be the total amount of time that machine Mj spends on job i in time period $I_1 = [0, d_1 + L_{max}]$. Furthermore, for k = 2, . . . , n let $t_{ij}^{(k)}$ be the total amount of time that machine M_j spends on job i within the time period $I_k = [d_{k-1} + L_{max}, d_k + L_{max}]$. Then we have to solve minimize L_{max} subject to

$$\Sigma_{j=1}^{m} \Sigma_{k=1}^{i} \frac{t_{ij}^{(k)}}{p_{ij}} = 1, \quad i=1,\ldots n$$

$$\Sigma_{j=1}^{m} \; t_{ij}^{(1)} \; \leq \; d_1 + L_{max}, \;\; i=1,...n$$

$$\Sigma_{j=1}^{m} \; t_{ij}^{(k)} \; \leq \; d_k - d_{k-1}, \;\; i=1,...n; \;\; k=2,...n$$

$$\Sigma_{i=1}^{n} \; t_{ij}^{(1)} \; \leq \; d_1 + L_{max}, \;\; j=1,...m$$

$$\Sigma_{i=k}^{n} \; t_{ij}^{(k)} \; \leq \; d_k - d_{k-1}, \;\; j=1,...m; \;\; k=2,...n$$

$$t_{ij}^{(k)} \geq 0, \quad j = 1,...,m; \; i, \; k = 1,...n.$$

Given an optimal solution of this linear programming problem, an L_{max} optimal schedule can be obtained by constructing for each of the time periods I_k ($k = 1, . . . , n$) a corresponding schedule using the data given by the matrix $T_k = (t_{ij}^{(k)})$. We again conclude that problem R|pmtn | L_{max} is polynomially solvable. In a similar way, we may solve problem R | pmtn; r_i | L_{max} by considering intervals $[t_k , t_{k+1}]$, $k = 1, . . . , r - 1$, where

$$t_1 < t_2 < . . . < t_r$$

is the ordered sequence of all r_i values and $d_i + L_{max}$ values. In this case, we have the variables $t_{ij}^{(k)}$ and L_{max} where $t_{ij}^{(k)}$ is the processing time of job i on M_j within the interval $[t_k , t_{k+1}]$ [2].

2.3Uniform Parallel Machines

We now consider n jobs J_i (i = 1, . . . , n) to be processed on m parallel uniform machines M_j (j = 1, . . . , m). The machines have different speeds s_j (j = 1, . . . , m) but the speed of each machine is constant and does not depend on the job. Every job J_i has a processing requirement p_i (i = 1, . . . , n). Execution of job J_i on machine M_j requires p_i / s_j time units. If we set $s_j = 1$ for j = 1, . . . , m. we have m parallel identical machines. All problems with parallel identical machines which are NP-hard are also NP-hard if we replace the machines by uniform machines. Therefore, we consider problems with preemptions. We also assume that $1 = s_1 \geq s_2 \geq \ldots \geq s_m$ and $p_1 \geq p_2 \geq \ldots \geq p_n$ [2].

2.3.1 Q | pmtn | C_{max}

Initially we will present a lower bound ω for the objective value of problem Q | pmtn |C_{max}. In the latter step, we will give an algorithm which constructs a schedule of length ω (i.e. an optimal schedule). Let $P_i = p_1 + \ldots + p_i$ and $S_j = s_1 + \ldots + s_j$ for i = 1, . . . , n and j = 1, . . . , m. Furthermore, we assume that $n \geq m$. If $n < m$, we only have to consider the n fastest machines. A necessary condition for processing all jobs in the interval [0, T] is

$$P_n = p_1 + \ldots + p_n \leq s_1 T + \ldots + s_m T = Sm\ T$$

or

$$P_n / S_m \leq T$$

Similarly,the condition $P_j / S_j \leq T$ should also be for j = 1, . . . , m−1 because P_j / S_j is a lower bound on the length of a schedule for the jobs $J_1, \dots, J_j$.

Thus,

$$\omega := max\{ max_{j=1}^{m-1} P_j / S_j ,\ P_n / S_m \}$$

is a lower bound for the C_{max} − values.

Now we will construct a schedule which achieves this bound. The corresponding algorithm is called the level algorithm. Given a partial schedule up to time t, the level $p_i(t)$ of job i at time t is the portion of p_i not processed before t. At time t, the level algorithm calls a procedure assign (t) which assigns jobs to machines. The machines run with this assignment until some time s > t. A new assignment is done at time s, and the process is repeated [2].

Algorithm level

1: t := 0;

2: WHILE there exist jobs with positive level DO

BEGIN

3: Assign(t);

4: t_1 := min{s > t | a job completes at time s };

5: t_2 := min{s > t | there are jobs i, j with $p_i(t) > p_j(t)$ and $p_i(s) = p_j(s)$ };

6: t:=min{t_1 , t_2 }

END

7: Construct the schedule.

The procedure assign(t) is given by

Assign (t)

1. J := { i | p_i (t) > 0 };

2. M := { M_1 , ... , M_m };

3. WHILE $J \neq \phi$ and $M \neq \phi$ DO

BEGIN

4. Find the set I ⊆ J of jobs with highest level;

5. r := min {| M |, | I |};

6. Assign jobs in I to be processed jointly on the r fastest machines in M;

7. $J := J \setminus I$

8. Eliminate the r fastest machines in M from M

END

The example with 5 jobs to be processed on 4 machines presented below in the figure shows how the algorithm works.

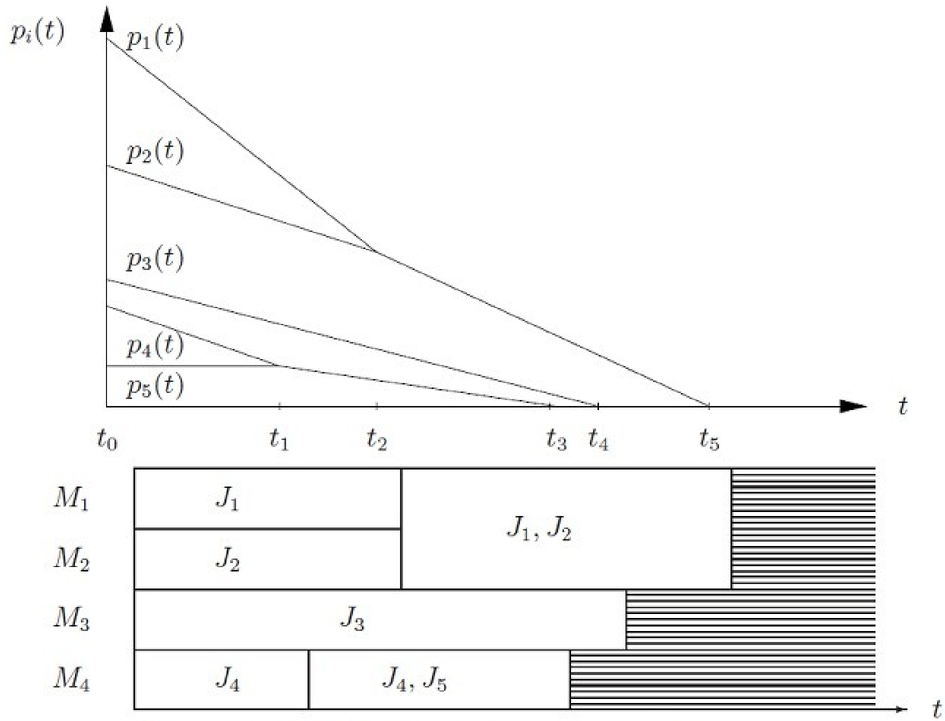

Figure 6 : Application of the level algorithm.

Initially, the four jobs 1,2,3,4 with the largest processing times are processed on machines M_1, M_2, M_3, M_4, respectively. At time t_1 job 4 has a level which is equal to the processing time of job 5. Thus, from time t_1 jobs 4 and 5 are processed jointly on machine M_4. Due to the fact that job 1 is processed on a faster machine than job 2 at time t_2, we reach the situation that $p_1(t_2) = p_2(t_2)$. Therefore, jobs 1 and 2 are processed jointly on both M_1 and M_2.

1	2	3	4	5	6
6	1	2	3	4	5
5	6	1	2	3	4

Figure 7 : Processing 6 jobs jointly on 3 machines.

To process r jobs 1, . . . , r jointly on l machines M_1 , . . . , M_l ($r \geq l$) during some time period T, we process each job during a period of T /r time units on each of the machines. A corresponding schedule is shown in the above figure (6 jobs 3 machines) for the case r = 6 and l = 3 [2].

CHAPTER 3

Application of the Level Algorithm

EXAMPLE 1: Consider the problem Q | pmtn |C_{max} with 5 jobs and 4 machines. Given are the processing times and speeds

$P_1 = 5$; $P_2 = 4$; $P_3 = 3$; $P_4 = 2$; $P_5 = 1$.

Harmonic progression is a progression formed by taking the reciprocals of an arithmetic progression. In other words, it is a sequence of the form
a , $\frac{a}{1+d}$, $\frac{a}{1+2d}$, $\frac{a}{1+3d}$ where -1/d is not a natural number.

(Note: Speeds are in a harmonic progression a=1 and d=1) .

$S_1 = 1$; $S_2 = \frac{1}{2}$; $S_3 = \frac{1}{3}$; $S_4 = \frac{1}{4}$.

Construct the optimal schedule using level algorithm and find the value of C_{max} ?

Solution: Initially we will present a lower bound ω for the objective value of problem Q | pmtn |C_{max}.

Let ' n ' be the number of jobs and ' m' be the number of machines. If n < m, we only have to consider the n fastest machines.

A necessary condition for processing all jobs in the interval [0,T] is

$$P_n / S_m < T \quad .$$

Similarly we must have $P_j/S_j<T$ for j=1,...,m-1 because P_j/S_j is a lower bound on the length of a schedule for the jobs . Thus $J_1, \ldots, J_j$.

$$\omega := max\{max_{j=1}^{m-1} P_j/S_j, P_n/S_m\}$$

is a lower bound for the C_{max} values.

$$P_n/S_m = (5+4+3+2+1)/1+0.50+0.33+0.25 \Rightarrow 15/2.08 \Rightarrow 7.2115$$

Similarly for P_j/S_j for j=1,...,m-1. Here m=4 so j = 1, 2, 3

$$\omega = max\{max\{(5/1),[(5+4)/(1+0.5)],[(5+4+3)/(1+0.50+0.33)]\}, 7.2115\};$$

$$\omega = max\{max\{5,6,6.55\}, 7.2115\}$$

$$\omega = 7.2115$$

We now plot the graph considering the jobs and speeds on Y- axis and X- axis respectively which results in the t values.

The slope of a line for a job i is considered to be the speed S_i .

The straight line equation for slope intercept form:

$$y = mx + b$$

b is the y-intercept and m is the slope.

To find the equation of line that passes through the point (5,0) with a slope of 1 for job J_1 is $y=-x+5$

Similarly we calculate the equation of line for jobs 2, 3 ,4 and 5 respectively.

$y=(-1/2)x+4;$

$y=(-1/3)x+3;$

$y=(-1/4)x+2;$

$y=1;$

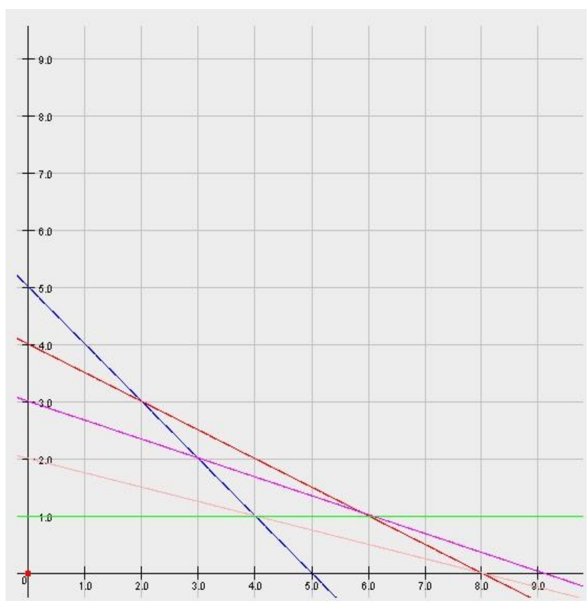

Figure 8 : Plotting the graph with processing times and speeds.

The first point of intersection is between job 1 and job 2 at (2 , 3) we get $t_1=2.0$. At this point of time job 1 and 2 are done jointly on machine 1and machine 2.

Re-plotting the graph with the new equations.

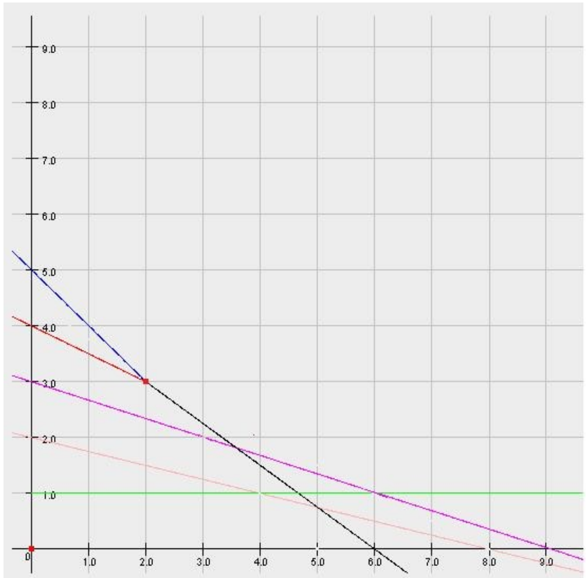

Figure 9 : Intersection of job 4 and job 5.

Similarly at t_2 job1, job 2 and job 3 intersect at (3.571, 1.821) the value of $\boldsymbol{t_2 = 3.571}$

At t_2 job 1 , job 2 and job 3 are done jointly on machine 1 ,2 and 3.

We now re-plot the graph with the jointly performed jobs J_1, J_2, J_3 .

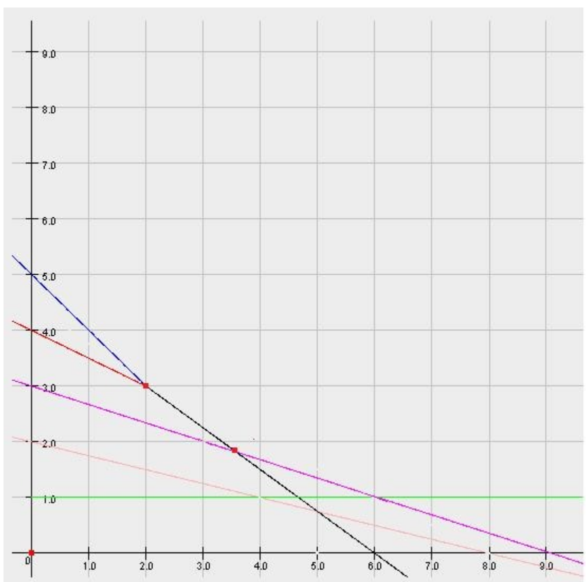

Figure 10: Intersection of job 3, job 4 and job 5.

t_3 is the point of time where job 4 and job 5 are done jointly on machine 4 the point of intersection of job 4 and job 5 is (4.0 ,1.0). The value of $\boldsymbol{t_3 = 4.0}$.

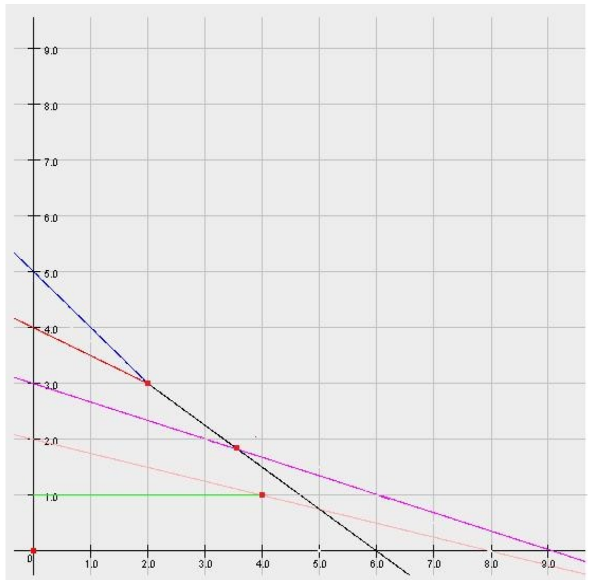

Figure 11: Intersection of job 1 and job 2.

t_4 is the point of intersections of job 1 , job 2 and job 3 with job 4 and job 5 i,e (5.422, 0.822). Hence the value of $\boldsymbol{t_4 = 5.422}$

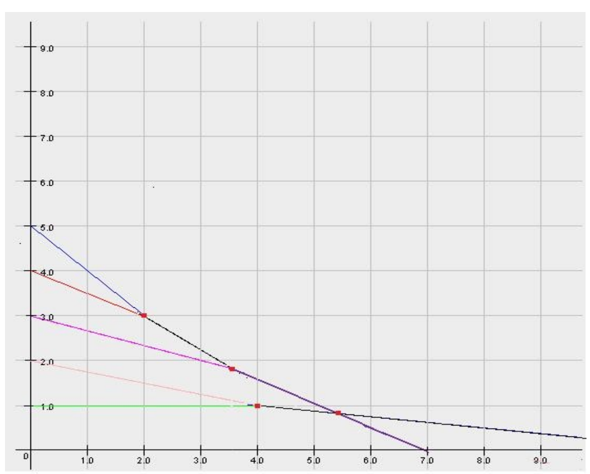

Figure 12 : Intersection of all jobs.

To calculate t_5 we re-plot the graph with the t_4 as the point of intersection of (1,2,3,4,5) and slope is considered to be the average of speeds of machines 1,2,3 and 4.

The value of $\boldsymbol{t_5 = 7.9}$

Final Graph is plotted with t_1, t_2, t_3, t_4 and t_5 .

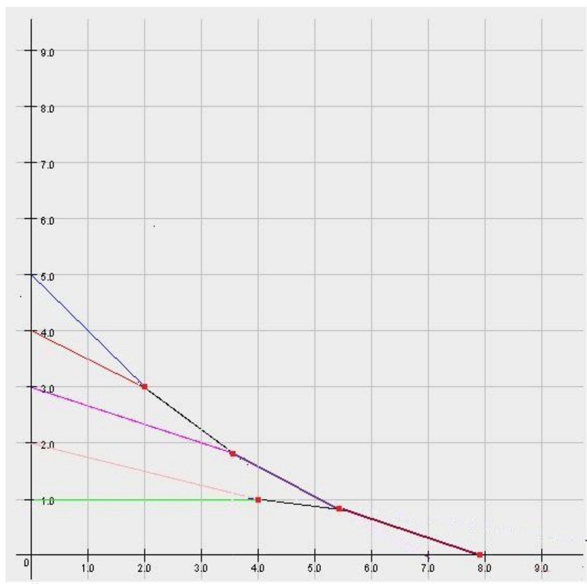

Figure 13 : Completion time of all jobs.

We now draw the optimal schedule for these jobs.

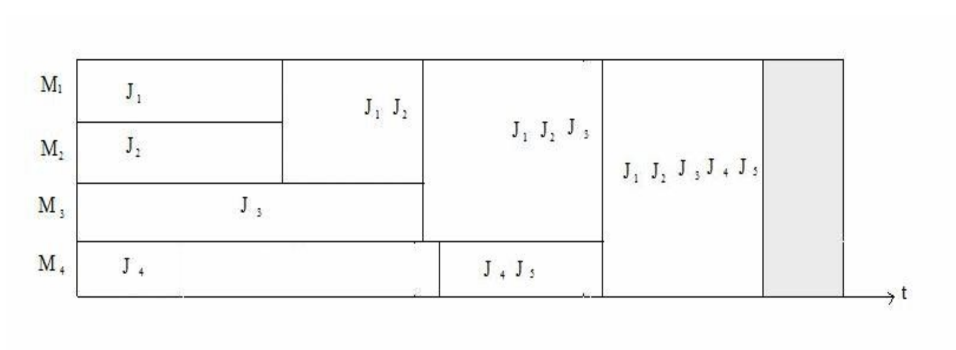

Figure 14: Optimal schedule for 5 jobs.

Example with same processing times but with different speeds.
Consider the problem Q | pmtn |C_{max} with 5 jobs on 4 machines.
Given are the processing times and speeds

$P_1=5$; $P_2=4$; $P_3=3$; $P_4=2$; $P_5=1$.

Harmonic progression is a progression formed by taking the reciprocals of an arithmetic progression. In other words, it is a sequence of the form
a , $\frac{a}{1+d}$, $\frac{a}{1+2d}$, $\frac{a}{1+3d}$ where -1/d is not a natural number.

Note: Speeds are in a harmonic progression with a=1 and d=0.5

$S_1=1$; $S_2=\frac{1}{(1+0.5)}\Rightarrow 0.666$; $S_3=\frac{1}{(1+1)}\Rightarrow 0.5$; $S_4=\frac{1}{(1+1.5)}\Rightarrow 0.4$.

Construct the optimal schedule using level algorithm and find the value of C_{max} .

Solution: Initially we will present a lower bound ω for the objective value of problem Q | pmtn | C_{max}.

Let ' n ' be the number of jobs and ' m ' be the number of machines. If $n<m$, we only have to consider the n fastest machines.

$$P_n/S_m<T$$

Similarly we must have $P_j/S_j<T$ for $j=1,\ldots,m-1$ because P_j/S_j is a lower bound on the length of a schedule for the jobs $J_1,\ldots,J_j$.

Thus

$$\omega:=max\{max_{j=1}^{m-1}P_j/S_j,P_n/S_m\}$$

is a lower bound for the C_{max} values.

$P_n/S_m = (5+4+3+2+1)/1+0.66+0.5+0.4 \Rightarrow 15/2.56 \Rightarrow 5.859375$

Similarly for P_j/S_j for $j=1,...m-1$

Here m=4 so j = 1, 2, 3

$\omega = max\{max(5/1),[(5+4)/(1+0.66)],[(5+4+3)/(1+0.66+0.50)], 5.859375¿;$

$\omega = max\{max\{5, 5.4216, 5.555\}, 5.8593\}$

$\omega = 5.8593$

We now plot the graph considering the jobs and speeds on Y- axis and X- axis respectively which results in the t values.

The slope of a line for a job i is considered to be the speed S_i .

The straight line equation for slope intercept form:

$y = mx + b$ where b is the y-intercept and m is the slope.

To find the equation of line that passes through the point (5,0) with a slope of 1 for job J_1 is

$y = -x + 5$

Similarly we calculate the equation of line for jobs 2, 3 ,4 and 5 respectively.

$y = -0.666x + 4;$

$y = -0.5x + 3;$

$y = -0.4x + 2;$

$y = 1;$

The resulting graph for the above plotted lines

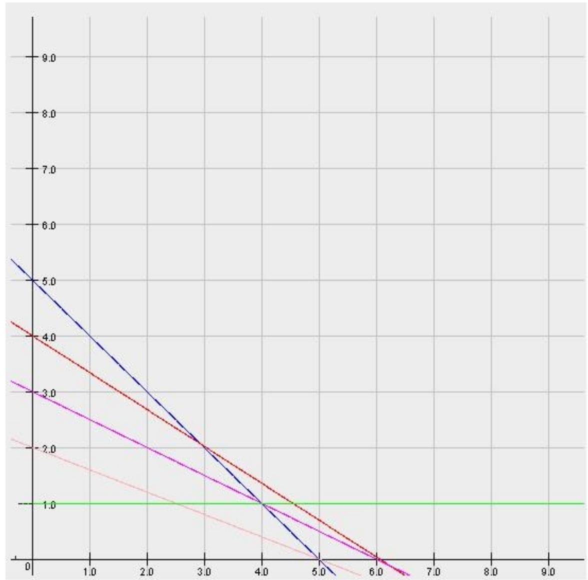

Figure 15: Plotting the graph with processing times and speeds.

The first point of intersection is between job 4 and job 5 at (2.5 , 1) we get $t_1 = 2.5$. After time t_1 job 4 and 5 merge and are processed jointly.

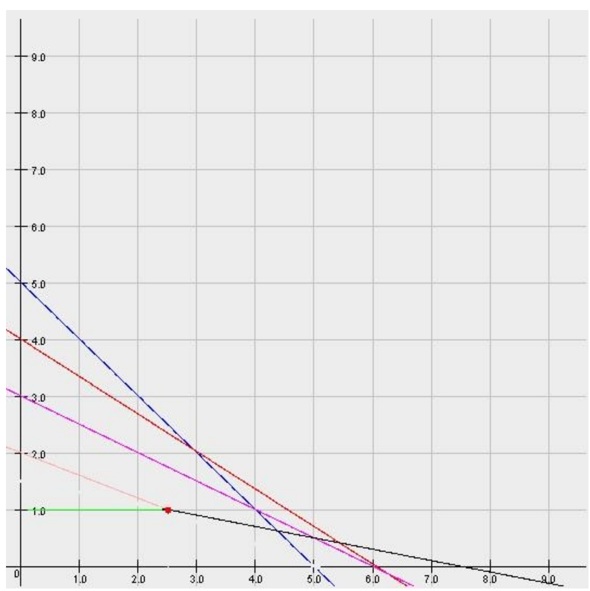

Figure 16: Intersection of job 4 and job 5.

Similarly at t_2 job1, job 2 intersect at (2.94, 2.05) the value of $\boldsymbol{t_2=2.94}$.

At t_2 job 1 and job 2 are done jointly on machine 1 and 2.

We now re-plot the graph with the jointly performed jobs J_1 and J_2

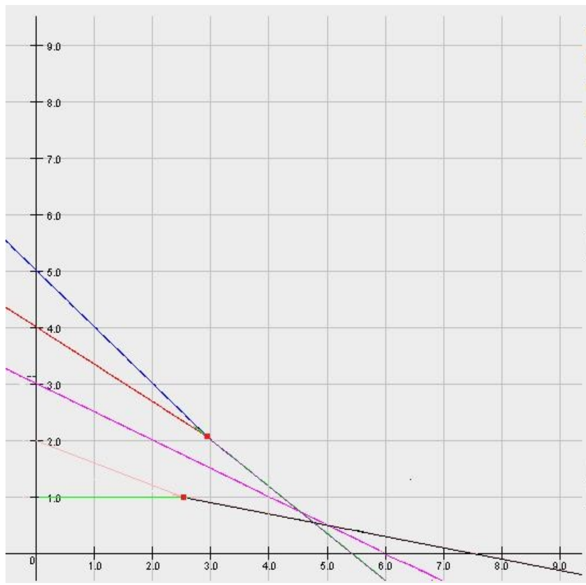

Figure 17 : Intersection of job 1 and job 2.

t_3 is the point of time where job 1,2 and job 3 are done jointly on machine 1, 2 and 3 the time (point)of intersection of all these jobs is (4.5454, 0.7272).The value of $\boldsymbol{t_3=4.5454}$.

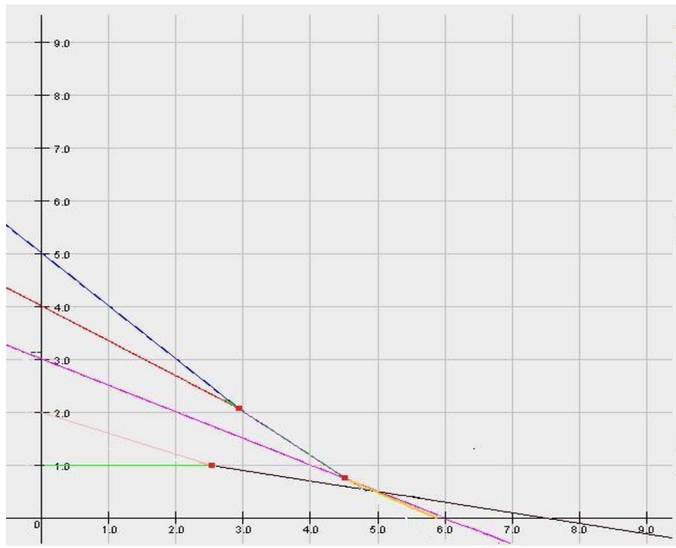

Figure 18 : Intersection of job 1 , job 2 and job 3.

At time t_4 job 1,2,3 and 4,5 intersect (4.761 , 0.547) and the value of

$\boldsymbol{t_4 = 4.761}$

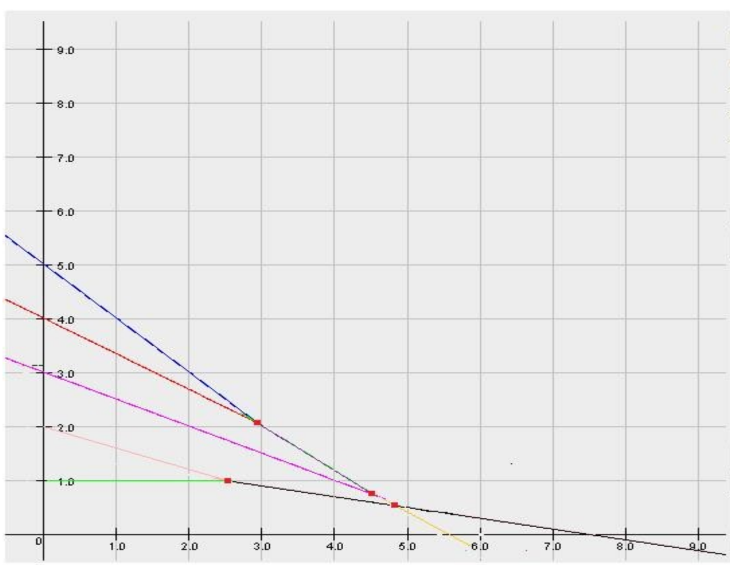

Figure 19 : Intersection of all jobs.

Job 1, 2 and Job 3 are combined with job 4 and 5 and are performed on Machines 1, 2, 3 and 4 and completed at 5.8

Hence the value of $t_5 = 5.8$.

Final Graph is plotted with t_1, t_2, t_3, t_4 and t_5 .

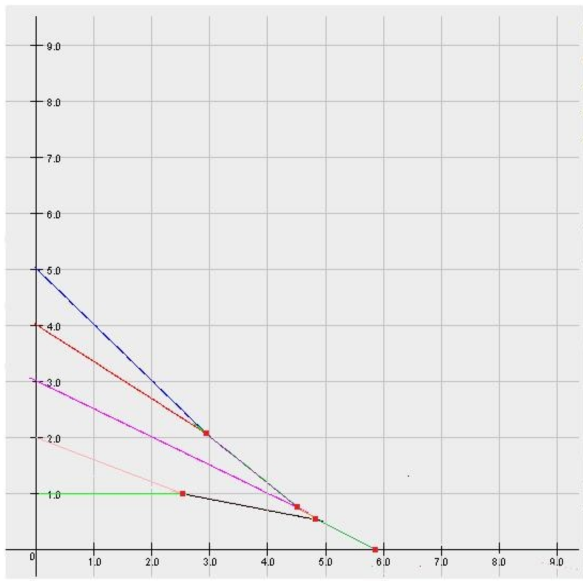

Figure 20 : Completion time of all jobs.

We now draw the optimal schedule for these jobs.

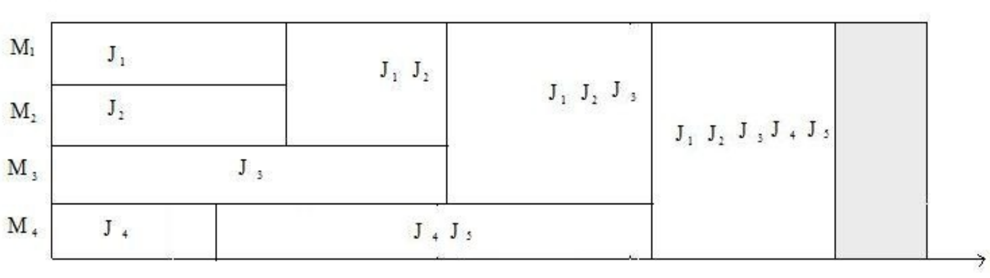

Figure 21 : Optimal schedule of 5 jobs.

The C_{max} value is max { 2.5, 2.94, 4.54, 4.76, 5.8} = 5.8

Theorem 2: Algorithm level constructs an optimal schedule for problem $Q|pmtn|C_{max}$ [2].

Proof **:**Because

$$\omega := max\{max_{j=1}^{m-1} P_j / S_j , P_n / S_m\}$$

is a lower bound for the schedule length, it is sufficient to show that the schedule constructed achieves this bound.

Assume that at the beginning of the level algorithm we have $p_1(0) \geq p_2(0) \geq ... \geq p_n(0)$. This order does not change during the algorithm, i.e. we have

$$p_1(0) \geq p_2(0) \geq \ldots \geq p_n(0) \quad \text{for all t.}$$

We assume that the algorithm always assigns jobs to machines in this order. To prove the desired property, we first assume that no machine is idle before all jobs are finished, say at time T. Then

$$T(s_1 + ... + s_m) = p_1 + p_2 + ... + p_n \quad \text{or} \quad T = P_n / S_m$$

Thus bound ω is achieved by the algorithm. If a machine is idle before the last job finishes, then for the finishing times $f_{1,} .., f_m$ of machines $M_1, ..., M_m$ we have

$$f_1 \geq f_2 \geq ... \geq f_m$$

Or Else, if $f_i < f_i + 1$ for some $1 \leq i \leq m-1$, the level of the last job processed on M_i at some time $f_i - \varepsilon$, where $\varepsilon > 0$ is sufficiently small, is smaller than the level of the last job on M_{i+1} at the same time. This is a contradiction. Furthermore, in the above equation we have at least one strict inequality.

Assume that $T := f_1 = f_2 = \ldots = f_j > f_{j+1}$ with $j < m$. The jobs finishing at time T must have been started at time 0. If this is not the case, then we have a job i which starts at time $t > 0$ and finishes at time T. This implies that at time 0 at least m jobs, say jobs $1, \ldots, m.$ are started and processed together on all machines. We have

$p_1(0) \geq \ldots \geq p_m(0) \geq p_i(0)$, which implies

$p_1(T-\varepsilon) \geq \ldots \geq p_m(T-\varepsilon) \geq p_i(T-\varepsilon) > 0$ for all ε with $0 < \varepsilon < T - t$.

Thus, until time T no machine is idle, which is a contradiction. We conclude $T = P_j / S_j$.

The level algorithm calls the procedure assign(t) at the most O(n) times. The computational effort for assigning jobs to machines after each call is bounded by O(nm). Thus, we get a total complexity of $O(n^2 m)$ (the total work for calculating all t values is dominated by this).

Theorem 3 : Given a set of parallel machines 'm ' with speeds in harmonic series and jobs 'n' with processing times all jobs complete together.

Instead of a formal proof we provide motivation:

We assume that n ≥ m and m = n-1. If n < m, we only have to consider the n fastest machines.

Similarly the speeds of the machines 'M ' are in harmonic series

a , $\frac{a}{1+d}$, $\frac{a}{1+2\mathrm{d}}$, $\frac{a}{1+3\mathrm{d}}$ where -1/d is not a natural number.

To prove that all jobs complete together we use that concept of divergent series.

One way to prove divergence is to compare the harmonic series with another divergent series:

$$\begin{aligned} & 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8} + \frac{1}{9} + \cdots \\ > \; & 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{16} + \cdots . \end{aligned}$$

Each term of the harmonic series is greater than or equal to the corresponding term of the second series, and therefore the sum of the harmonic series must be greater than the sum of the second series. However, the sum of the second series is infinite:

$$1 + \left(\frac{1}{2}\right) + \left(\frac{1}{4} + \frac{1}{4}\right) + \left(\frac{1}{8} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8}\right) + \left(\frac{1}{16} + \cdots + \frac{1}{16}\right) + \cdots$$

$$= 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \cdots = \infty.$$

It follows that the sum of the harmonic series must be infinite as well.

More precisely, the comparison above proves that

$$\sum_{n=1}^{2^k} \frac{1}{n} \geq 1 + \frac{k}{2}$$ for every positive integer *k*

It can also be proved by the integral test that harmonic series diverges very slowly.

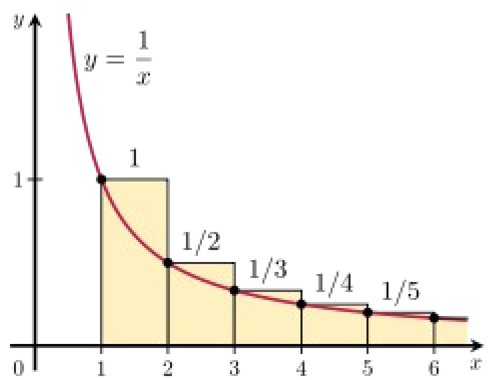

Figure 22 : Harmonic series diverges.

Harmonic series have terms that overlap with the adjacent term there by diverging.

Using the level algorithm and obtaining an optimal schedule with speeds in harmonic progression we observe that the optimal schedule leads to the completion of all jobs at the same time.

CHAPTER 4

Slot Scheduling Theory

As discussed in Chapter 3, the level algorithm produces an optimal schedule. This chapter is divided in two sections. We first discuss the ad-slot scheduling mechanism. In the second part we discuss the application of the level algorithm in Internet ad-slot placement.

4.1Ad-slot scheduling

One of the more visible means by which the Internet has disrupted traditional activity is the manner in which advertising is sold. Offline, the price for advertising is typically set by negotiation or posted price. Online, much advertising is sold via auction. Most prominently, Web search engines like Google and Yahoo! auction space next to search results, a practice known as sponsored search.

Sponsored search is a form of advertising typically sold at auction where merchants bid for positioning along side web search results. Web search engines monetize their service by auctioning off advertising space next to their standard algorithmic search results [27]. For example, Pepsi or sunkist may bid to appear among the advertisements usually located above or to the right of the algorithmic results whenever users search for “*soda* “.

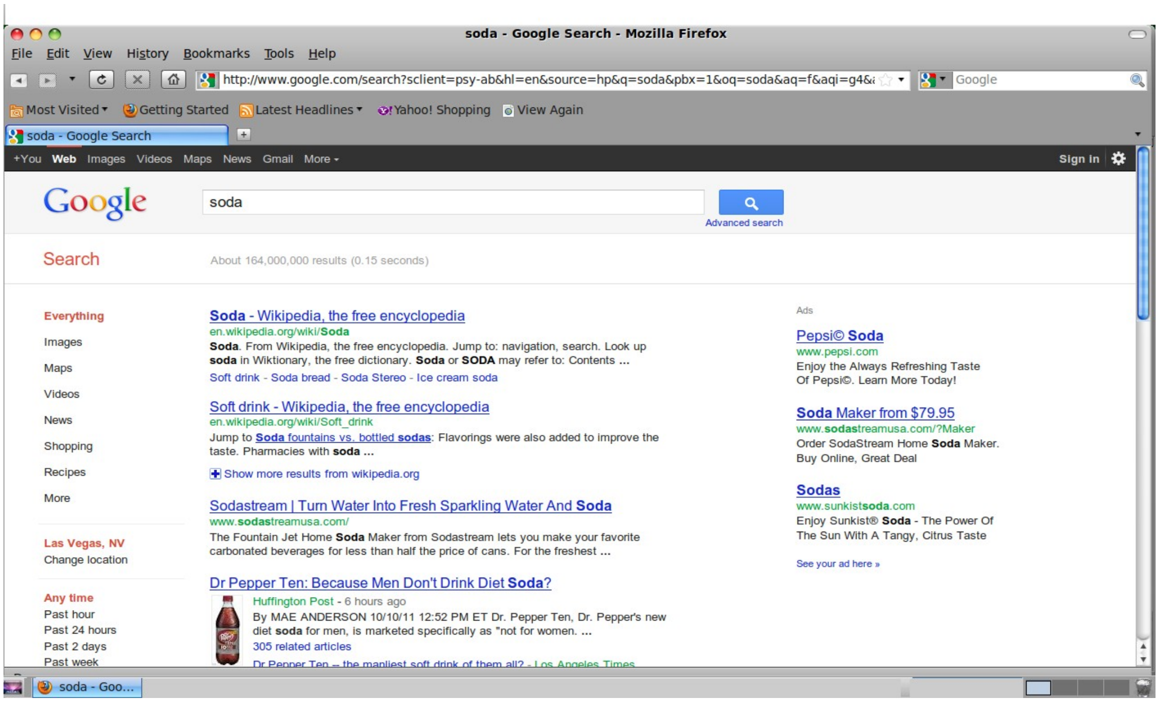

Figure 23 : Screen shot of user query with the search results on the left and the ads on the right.

These sponsored results are displayed in a format similar to algorithmic results: as a list of items each containing a title, a text description, and a hyperlink to the advertiser's Web page. We call each position in the list a slot.

Basically, there are three parties involved in sponsored search[22].

• The first party is the advertisers who have multiple objectives in seeking to place advertisements. Some advertisers want to develop their brand, some seek to make sales, and yet others advertise for defensive purposes on specific keywords central to their business. Some have budget constraints, while others are willing to spend as much as it takes to achieve their goal. Some seek to obtain many clicks and eyeballs, yet others attempt to optimize their return on investment. So, in general, advertisers are of varied types [22].

• The second party is the auctioneer, in this case, the search engine. The search engines have to balance many needs. They must maintain useful search results and have advertisements enhance, rather than interfere with, the search experience. They need to make sure the advertisers get their needs fulfilled, and at the same time ensure that the market the advertisers participate in is efficient and conducive to business.

• The third party is perhaps the most important in the game: these are search users. Users come to search engines for information and pointers. In addition, they also come to discover shopping opportunities, good deals, and new products. There are millions of users with different goals and behavior patterns with respect to advertisements [22].

Ad slot is a premium ad sales platform used by publishers to increase revenue and significantly reduce cost of sales. The process of choosing and charging the advertisers is a daunting algorithmic and engineering task. The search engines typically take in to consideration several factors including the search key word, the demographics of the user, the frequency of the keyword, as well as the bid, budget and click through rate of the advertisers for each of these decisions.

We consider the Ad Slot Scheduling problem, where advertisers must be scheduled to sponsored search slots during a given period of time. Advertisers specify a budget constraint, as well as a maximum cost per click, and may not be assigned to more than one slot for a particular search [5].

A natural mechanism for Ad Slot Scheduling is the following: Find a feasible schedule and a set of prices that maximizes revenue, subject to the bidders' constraints. It is straightforward to derive a linear program for this optimization problem, but unfortunately this is not a truthful mechanism. However, there is a direct truthful mechanism—the price-setting mechanism that results in the same outcome as an equilibrium of the revenue-maximizing mechanism.

Jon et al. [5] derive this mechanism (and prove that it is truthful) by starting with the single-slot case, where two extreme cases have natural, instructive interpretations. With only bids (and unlimited budgets), a winner-take-all mechanism works; with only budgets (and unlimited bids) the clicks are simply divided up in proportion to budgets. Combining these ideas in the right way results in a natural descending-price mechanism, where the price (per click) stops at the point where the bidders who can afford that price have enough budget to purchase all of the clicks.

Generalizing to multiple slots requires understanding the structure of feasible schedules, even in the special budgets-only case. We solve the budgets-only case by characterizing the allowable schedules in terms of the solution (level algorithm) to the problem of Q | pmtn | C_{max}. The difficulty that arises is that the lengths of the jobs in the scheduling problem actually depend on the price charged. Thus, we in corporate the scheduling algorithm into a descending-price mechanism, where the price stops at the point where the scheduling constraints are tight; at this point a block of slots is allocated at a fixed uniform price (dividing the clicks equally by budget) and the mechanism iterates.

4.2 Single slot

In this section we consider only one advertising slot with some number of clicks. As mentioned earlier we consider two cases single slot with budgets only and single slot with bids and budgets. We represent the bids as $b_{1,}\ldots,b_n$, budgets as $B_1,\ldots,B_n$ and ' D ' as the number of clicks.

4.2.1 Single-slot with budgets-only

Our input in this case is a set of budgets $B_1,\ldots,B_n$, and consider all bids as $b_i=\infty$ we are supposed to allocate D clicks with no ceiling on the per-click price. We apply the principle of *Proportional sharing* (**Proportional Share Scheduling** is a type of scheduling which preallocates certain amount of time to each of the processes). Let

$B = \sum_i B_i$. Now to each bidder i, allocate $(B_i / B)D$ clicks. Set all prices the same: $p_i = p = B/D$. The mechanism guarantees that each bidder exactly spends his/her budget, thus no bidder will report $B_i' > B_i$. Now suppose some bidder reports $B_i' = B_i - \Delta$, for $\Delta > 0$. Then this bidder is allocated $D(B_i - \Delta) / (B - \Delta)$ clicks, which is less than $D(B_i / B)$, since $n > 1$ and all $B_i > 0$ [5] [22].

Example 1: Suppose there are three bidders and D = 100 clicks in a single slot. Bidder 1 has a budget B_1=\$25 , bidder 2 has B_2 = \$15 and bidder 3 has B_3= \$10. Allocate the number of clicks to each bidder.

Figure 24 : SIngle slot ; D clicks

Solution: Let us calculate $B = \sum_i B_i$

$$B = 25+15+10 = 50.$$

The price for all bidders is $p = B / D = 50 / 100 \Rightarrow 0.5$

Allocating the number of clicks for bidder 1 $c_1 = D * (B_1 / B)$

$$c_1 = 100 * (25 / 50)$$

$$c_1 = 50 \text{ clicks.}$$

Similarly, allocating the number of clicks for bidder 2 $c_2 = D * (B_2 / B)$

$$c_2 = 100 * (15 / 50)$$

$c_2 = 30$ clicks.

Allocating the number of clicks for bidder 3 $c_3 = D *(B_3 / B)$

$c_3 = 100 * (10 /50)$

$c_3 = 20$ clicks.

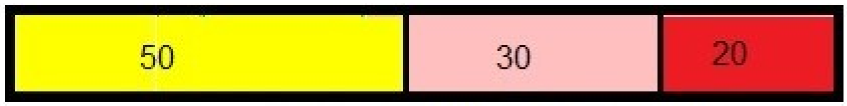

Figure 25: Allocation of D clicks.

4.2.2 Single-slot with bids and budgets.

Let us first assume all budgets $B_i = \infty$. Then, our input amounts to bids $b_1 > b_2 > . . . > b_n$. The obvious mechanism is simply to give all the clicks to the highest bidder. We charge bidder 1 her full price $p_1 = b_1$. A simple argument shows that reporting the truth is a weakly dominant strategy for this mechanism. The losing bidders cannot gain from decreasing b_i. The winning bidder can lower her price by lowering b_i, but this will not gain her any more clicks, since she is already getting all D of them. We incorporate the price setting mechanism essentially the descending price mechanism: the price stops descending when the bidders willing to pay at that price have enough budget to purchase all the clicks. We have to be careful at the moment a bidder is added to the pool of the willing bidders; if this new bidder has a large enough budget, then suddenly the willing bidders have more than enough budget to pay for all of the clicks. To compensate, the mechanism decreases this “threshold” bidder’s effective budget until the clicks are paid for exactly.

Price-Setting (PS) Mechanism (Single Slot with bids and budgets)

- Assume wlog that $b_1 > b_2 > \ldots > b_n \geq 0$.
- Let k be the first bidder such that $b_{k+1} \leq \sum_{i=1}^{k} B_i / D$. Compute price $p = min\{\sum_{i=1}^{k} B_i / D, b_k\}$.
- Allocate B_i / p clicks to each $i \leq k-1$. Allocate $\hat{B}_k / p$ clicks to bidder k, where $\hat{B}_k = pD - \sum_{i=1}^{k-1} B_i$.

Example 2 : Suppose there are four bidders with b_1 = \$3, b_2 = \$2, b_3 = \$1, b_4 = \$0.25 and B_1 = \$20, B_2 = \$60, B_3 = \$40, B_4 = \$5 and D = 100 clicks. Allocate appropriate clicks based on the price-setting mechanism.

Solution: In this case $b_1 > b_2 > \ldots > b_n \geq 0$

Case 1: Let k=1 be the first bidder and lets check for the condition $b_{k+1} \leq \sum_{i=1}^{k} B_i / D$

$$b_{1+1} \leq \sum_{i=1}^{1} B_i / 100$$

$$b_2 \leq 20/100$$

Let us substitute the value of b_2 we get $2 \leq 0.2$ This condition does not satisfy.

Case 2 :Let k=2 be the first bidder and lets check for the condition

$b_{k+1} \le \sum_{i=1}^{k} B_i / D$

$$b_{2+1} \le \sum_{i=1}^{2} B_i / 100$$

$$b_3 \le (20+60)/100$$

Let us substitute the value of b_3 we get $1 \le 0.8$ This condition does not satisfy.

Case 3 :Let k=3 be the first bidder and lets check for the condition

$b_{k+1} \le \sum_{i=1}^{k} B_i / D$

$$b_{3+1} \le \sum_{i=1}^{3} B_i / 100$$

$$b_4 \le (20+60+40)/100$$

Let us substitute the value of b_4 we get $0.25 \le 1.2$ This condition satisfies.

Running the PS mechanism we get k = 3

The price is then set as $p = min\{\sum_{i=1}^{k} B_i / D, b_k\}$

$$p = min\{\sum_{i=1}^{3} B_i / D, b_3\}$$

$$p = min\{\frac{(20+60+40)}{100}, 1\}$$

$$p = 1$$

Allocating B_i / p clicks to each $i \le k-1$ we get $i \le 2 \, as \, k = 3$

When i=1 ; 20/1 => 20 clicks are allocated to bidder 1.

When i=2 ; 60/1 => 60 clicks are allocated to bidder 2.

Remaining clicks are allocated based on $\hat{B}_k / p$ clicks to bidder k, where $\hat{B}_k = pD - \sum_{i=1}^{k-1} B_i$ as per the price setting mechanism.

$\hat{B}_k = pD - \sum_{i=1}^{k-1} B_i$ Here k=3 , p=1, and D=100

Hence $\hat{B}_3 = (1 * 100) - (20 + 60)$ we get $\hat{B}_3 = 20$

Figure 26 : Single slot with Budgets and Bidders 1,2 and 3

Therefore bidder 1 gets 20 clicks, bidder 2 gets 60 clicks and bidder 3 gets 20 clicks and only $20 of bidder 3 budget is used. There is no threshold bidder.

4.3 Multiple Slots

Generalizing to multiple slots makes the scheduling constraint nontrivial. Now instead of splitting a pool of D clicks arbitrarily, we need to assign clicks that correspond to a feasible schedule of bidders to slots. The conditions under which this is possible add a complexity that needs to be incorporated into the mechanism.

As in the single-slot case it will be instructive to consider first the cases of infinite bids or budgets. Suppose all $B_i = \infty$. In this case, the input consists of bids only $b_1 > b_2 > \ldots > b_n$. Naturally, what we do here is rank by bid, and allocate the slots to the bidders in that order. Since each budget is infinite, we can always set the prices p_i equal to the bids b_i. By the same logic as in the single-slot case, this is easily seen to be truthful. In the other case, when $b_i = \infty$, there is a lot more work to do.

Without loss of generality, we may assume the number of slots equals the number of bids (i.e., n′ = n); if this is not the case, then we add dummy bidders with $B_i = b_i = 0$, or dummy slots with $D_i = 0$, as appropriate.

Assigning slots using a classical scheduling algorithm:

First we give an important lemma that characterizes the conditions under which a set of bidders can be allocated to a set of slots, which turns out to be just a restatement of a classical result from scheduling theory.

Lemma 1 [5][22] : Suppose we would like to assign an arbitrary set {1, . . . , k} of bidders to a set of slots {1, . . . , k} with $D_1 > \ldots > D_k$. Then, a click allocation $c_1 \geq \ldots \geq c_k$ is feasible iff

$$c_1 + \ldots + c_k \leq D_1 + \ldots + D_k \; for\ all\ l = 1, \ldots, k.$$

Proof: In scheduling theory, we say a job with service requirement x is a task that needs x/s units of time to complete on a machine with speed s. The question of whether there is a feasible allocation is equivalent to the following scheduling problem: Given k jobs with service requirements $x_i = c_i$, and k machines with speeds $s_i = D_i$, there a schedule of jobs to machines (with preemption allowed) that completes in one unit of time ?

As shown in Chapter 3 the optimal schedule for this problem

(a.k.a. Q | pmtn | C_{max}) can be found efficiently by the level algorithm,Level algorithm and the schedule completes in time $max_{l \leq k} \sum_{i=1}^{l} x_i / \sum_{i=1}^{l} s_i$. Thus, the conditions of the lemma are exactly the conditions under which the schedule completes in one unit of time.

4.3.1 Multiple-Slot Budgets-only

This mechanism is roughly a Descending-price mechanism where we decrease the price until a prefix of budgets fits tightly into a prefix of positions at that price, where upon we allocate that prefix, and continue to decrease the price for the remaining bidders. More formally, it can be written as follows [5] [22]:

Price-Setting Mechanism (Multiple Slots, Budgets Only)

- If all $D_i = 0$, assign bidders to slots arbitrarily and exit.
- Sort the bidders by budget and assume wlog that $B_1 \geq B_2 \geq \ldots \geq B_n$.
- Define $r_l = \sum_{i=1}^{l} B_i / \sum_{i=1}^{l} D_i$. Set price $p = max_l r_l$.
- Let l^* be the largest l such that $r_l = p$. Allocate slots $\{1, \ldots l^*\}$ to bidders $\{1, \ldots l^*\}$ at price p, using all of their budgets; i.e., $c_i = B_i / p$.
- Repeat the steps above on the remaining bidders and slots until all slots are allocated.

Example of Multiple-Slot with Budgets-only :

Suppose there are four bidders A,B,C and D with B_1 = \$80, B_2 = \$70, B_3 = \$20, B_4 = \$1 and D_1 = 100, D_2 = 50, D_3 = 25, and D_4 =0. Allocate appropriate clicks based on the price-setting mechanism for multiple slots with budgets-only.

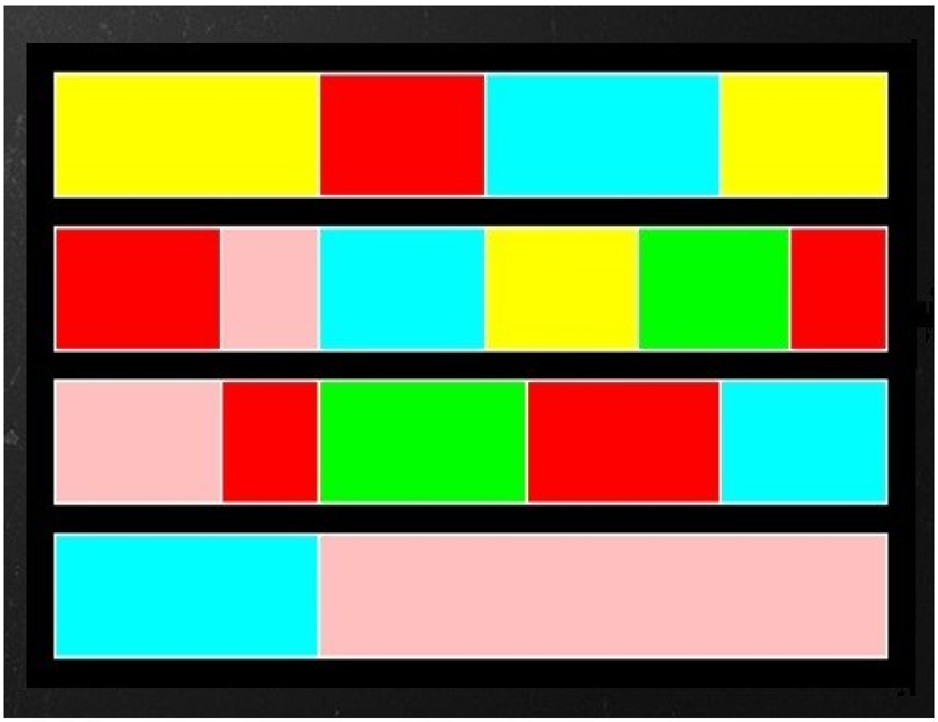

Figure 27 : Multiple slots

Solution: In the example $D_i \neq 0$ so we cannot assign bidders to slot arbitrarily and exit.

We then sort the bidders by budget , but we do not need to sort as they are already sorted in the order $B_1 \geq B_2 \geq \ldots \geq B_n$

For l=1 $r_l = \sum_{i=1}^{l} B_i / \sum_{i=1}^{l} D_i$ the value of $r_1 = (80/100)$ Therefore $r_1 = 0.8$

For l=2 $r_l=\sum_{i=1}^{l} B_i / \sum_{i=1}^{l} D_i$ the value of $r_2=[(80+70)/(100+50)]$

Therefore $r_2=1$

For l=3 $r_l=\sum_{i=1}^{l} B_i / \sum_{i=1}^{l} D_i$ the value of $r_3=[(80+70+20)/(100+50+25)]$

Therefore $r_3=0.971$

For l=4 $r_l=\sum_{i=1}^{l} B_i / \sum_{i=1}^{l} D_i$ the value of $r_4=[(80+70+20+1)/(100+50+25)]$

Therefore $r_4=0.977$

Here $l^* = 2$ since the largest values among r is r_2

Allocate slots $\{1, \ldots 2\}$ to bidders $\{1, \ldots 2\}$ at a price p=1 , using all of their budgets; i.e., $c_i=B_i / p$

$c_1=B_1/p$ Therefore the no. of clicks $c_1=80/1$ => $c_1=80$

Similarly $c_2=B_2/p$ Therefore the no. of clicks $c_2=70/1$ => $c_2=70$

We similarly repeat the above steps on the remaining bidders and slots until all slots are allocated.

In the second price block, we get $B_3/D_3=20/25$ and $(B_3+B_4)/(D_3+D_4)=21/25$. Thus p_2 is set to 21/25 = $0.84,

Bidder 3 gets 500/21 (approx 24) clicks and bidder 4 gets 25/21 (approximately 1) click, using the schedule as shown.

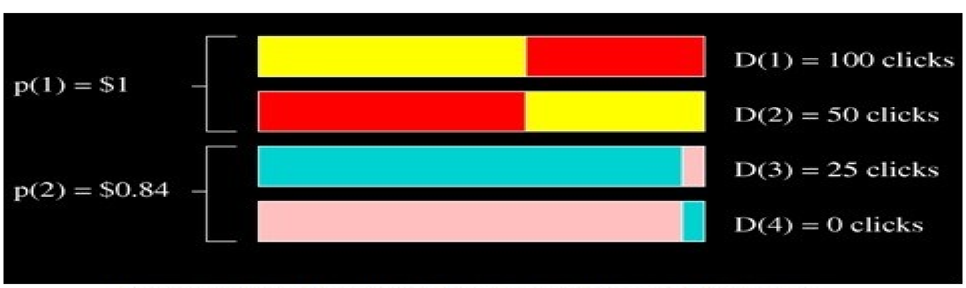

Figure 28 : Allocation of multiple slots with budgets

4.3.2 Multiple-Slots with bids and budgets

The generalization of the multiple slot price setting mechanism to use both bids and budgets combines the ideas from the bids and-budgets version of the single slot mechanism with the budgets-only version of the multiple-slot mechanism. As our price descends, we maintain a set of “active” bidders with bids at or above this price, as in the single-slot mechanism. These active bidders are kept ranked by budget, and when the price reaches the point where a prefix of bidders fits into a prefix of slots (as in the budgets-only mechanism) we allocate them and repeat. As in the single-slot case, we must be careful when a bidder enters the active set and suddenly causes an over-fit; in this case we again reduce the budget of this “threshold” bidder until it fits [5][22].

Price-setting Mechanism (Multiple slot with Bids and Budgets)

• Assume wlog that $b_1 > b_2 > \ldots > b_n = 0.$
• Let k be the first bidder such that running price-setting mechanism on bidders $1, \ldots k$ .would result in a price $p \geq b_{k+1}$.
• Reduce B_k until running price-setting mechanism on bidders $1, \ldots, k$ would result in a price $p \leq b_k$. Apply this allocation, which for some g $l^* \leq k$ ives the first l^* slots to the l^* bidders among $1, \ldots, k$ with the largest budgets.
• Repeat the above steps on the remaining bidders and slots until all slots are allocated.

Example for multiple-slots with bids and budgets.

Suppose there are four bidders A,B,C and D with B_1 = \$80, B_2 = \$70, B_3 = \$20, B_4 = \$1 and D_1 = 100, D_2 = 50, D_3 = 25, and D_4 =0 . Bids are also assigned for each bidder b_1 = \$3 , b_2 = \$0.75, b_3 = \$1, b_4 = \$0.50. Allocate appropriate clicks based on the price-setting mechanism for multiple slots with bids and budgets.

Solution : As per the assumption of w log we are supposed to have

$b_1 > b_2 > ... > b_n = 0.$

We first re arrange the bids which leads to b_2 = \$1, b_3 = \$0.75.

Running Price-Block mechanism on only bidder 1 gives a price of

$r_1 = 80/100$,

0.8 which is less than the next bid of \$1.

So, we re-run Price-Block mechanism on bidders 1 and 3 (the next-highest bid), giving $r_1 = 80/100$ and $r_2 = 100/150$.

We still get a price of \$0.80, but now this is more than the next-highest bid of \$0.75, so we allocate the first bidder to the first slot at a price of \$0.80. We are left with bidders 2-4 and slots 2-4. With just bidder 3 (the highest bidder) and slot 2, we get a price p = 20/50 => 0.4

0.4 which is less than the next-highest bid of \$0.75, so we consider bidders 2 and 3 on slots 2 and 3.

This gives a price of max{70/50, 90/75} = \$1.40, which is more than \$0.50. Since this is also more than \$0.75, we must lower B_2 until the price is exactly \$0.75, which makes B_2 = \$36.25.

With this setting of B_2 , Price setting allocates bidders 2 and 3 to slots 2 and 3, giving 75(36.25/56.25) and 75(20/56.25) clicks respectively, at a price of $0.75 per click.

Bidder 4 is allocated to slot 4, receiving zero clicks.

Note that by the same logic as the budgets-only mechanism, the prices $p_1, p_2, \ldots$ for each price block strictly decreases.

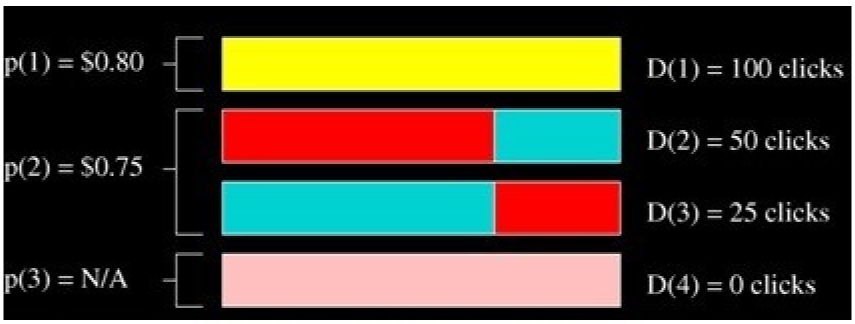

Figure 29 : Allocation of multiple slots with Bids and Budgets.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this paper we studied the types of parallel machines in particular we have illustrated examples pertaining to uniform parallel machines and its application in the real world. We have performed various experiments based on the level algorithm and have tried to present the behavior of jobs based on the speeds of machines. When the speeds relating to the machines and processing times were considered to be in a harmonic progression the completion time of all jobs was same.

We have presented a existing mechanism that involves assigning of bidders to the slots based on the classical result from scheduling theory to characterize the possible allocations. The algorithmic approach was taken in to consideration when allocation of ad slots was done based on the level algorithm which is polynomially solvable. As bidders get added in price setting mechanism, maintaining a sorted list of bidders and budgets can be done in time O(n log n). Thus it remains to show that it can be done in O(n) time given these sorted lists. Computing the ratios r_l and allocation can also be done in linear time.

This thesis focuses on technical preview of ad slot scheduling in generating a maximum revenue based on bidders, budgets and slots.

But there could also be many constraints that could improve the quality, efficiency and revenue of the ad slot system which include user click behavior, number and size of slots, advertiser weights.

Bidders can be provided with incentives like payment schemes, refunds and cancellations. An additional aspect of the problem from the auctioneer's perspective is how to target ads, that is, how to choose the keywords from the surrounding context. Consequently, the the resulting algorithmic approach to revenue maximizing of ad slot scheduling is more intricate and largely unexplored.

BIBLIOGRAPHY

1. Anis Gharbi., Exact Algorithms for Scheduling Parallel Machines with Heads and Tails.

2. Peter Brucker., Scheduling algorithms. Fifth edition. Springer-Verlag, Guildford, Surrey: 2004.

3. Martin Gairing, Burkhard Monien, and Andreas Woclaw., A Faster Combinatorial Approximation Algorithm for Scheduling Unrelated Parallel Machines.

4. Zhi-Long Chen., Department of Systems Engineering, University of Pennsylvania Philadelphia, PA 19104-6315. Solving Parallel Machines Scheduling Problems by Column generation.

5. Jon Feldman, S. Muthukrishnan, Evdokia Nikolova, and Martin Pal., A Truthful Mechanism for Offline Ad Slot Scheduling.

6. J.K Lenstra, A.H.G Rinnoy Kan., An introduction to multi-processor scheduling.

7. Joseph Y-T. Leung., Handbook of scheduling: algorithms, models, and performance analysis, Chapter 3 and Chapter 9.

8. E. L. Lawler (1979 a)., Preemptive scheduling of uniform parallel machines to minimize the weighted number of late jobs, Report BW 105, Center for Mathematics and Computer Science, Amsterdam, Netherlands, 1979.

9. Cheng and Sin (1990)., Parallel Machine Scheduling.

10. Chen, Potts and Woeginger (1998)., A review of Machine scheduling.

11. Wenxun Xing, Jiawei Zhang., Parallel machine scheduling with splitting jobs.Discrete Applied Mathematics 103 (2000) 259–269.

12. Tom Ebenlendr, Ji Sgall., Semi-Online Preemptive Scheduling: One Algorithm for All Variants.

13. Lirong Xia, Vincent Conitzer, Ariel D.Procaccia., A scheduling approach to coalitional manipulation, Proceedings of the 11th ACM conference on Electronic commerce, June 07-11, 2010, Cambridge, Massachusetts, USA.

[doi>10.1145/1807342.1807386]

14. Dell Amico, M. and Martello S., Optimal scheduling of tasks on identical parallel processors, ORSA Journal on Computing 7 (1995), 191–200.

15. Horvath, E.C., Lam, S., and Sethi, R., A level algorithm for preemptive scheduling, Journal of the ACM 24 (1977), 32–43.

16. Gonzalez, T. and Sahni, S., Preemptive scheduling of uniform processor systems, Journal of the ACM 25 (1978), 92–101.

17. Lawler, E.L. and Labetoulle, J., On preemptive scheduling of unrelated parallel processors by linear programming, Journal of the ACM 25 (1978), 612–619.

18. McNaughton., R Scheduling with deadlines loss functions Manage Sct 12, 1 (Oct 1959), 1-12.

19. Muntz, R R, and Coffman, E G JR., Optimal preemptive scheduling on two-processor systems IEEE Trans Computers C-18, 11 (Nov 1969), 1014-1020.

20. Muntz, R R, and Coffman, E G JR., Preemptive scheduling of real Ume tasks on multiprocessor systems J ACM 17, 2 (April 1970), 324-338.

21. Oliver Braun and Günter Schmidt., Parallel processor scheduling with limited number of preemptions.

22. Jon Feldman, S. Muthukrishnan., Algorithmic Methods for Sponsored Search Advertising, *Performance Modeling and Engineering (Proc. SIGMETRICS 2008 Tutorial Sessions)*,pp.91-124. [**www1.cs.columbia.edu**]

23. Gagan Aggarwal, Jon Feldman, Martin Pal, S. Muthukrishnan., Sponsored Search Auctions for Markovian Users, *Fourth Workshop on Ad Auctions; Workshop on Internet and Network Economics (WINE).*, 2008.

24. Ashish Goel, Mohammad Mahdian, Hamid Nazerzadeh, Amin Saberi., Advertisement Allocation for Generalized Second Pricing Schemes.

25. Benjamin Edelman, Michael Ostrovsky, Michael Schwarz, Thank Drew Fudenberg, Louis Kaplow, Robin Lee, Paul Milgrom, Muriel Niederle, Ariel Pakes., Internet Advertising and the Generalized Second Price Auction: Selling Billions of Dollars Worth of Keywords (2005).

26.Gagan Aggarwal, S. Muthukrishnan, David Pal, Martin Pal., General auction mechanism for search advertising, Proceedings of the 18th international conference on World wide web, April 20-24, 2009, Madrid, Spain. [doi>10.1145/1526709.1526742]

27. Noam Nisan., Algorithmic Game Theory, Chapter 28.

28. Equation of a line., http://www.webmath.com/equline1.html;

29. Plotting Graphs., http://graph.seriesmathstudy.com/

VITA

Graduate College

University of Nevada, Las Vegas

Shaista Lubna

Degrees:

Bachelor of Engineering, Computer Science and Information Technology, 2006

JNTU University

Master of Science, Computer Science, 2011

University of Nevada, Las Vegas

Thesis Title: Parallel Machines scheduling with applications to Internet ad-slot placement.

Thesis Examination Committee:

Chairperson, Dr. Wolfgang Bein, Ph.D.

Committee Member, Dr. Ajoy K Datta, Ph.D.

Committee Member, Dr. Lawrence Larmore, Ph.D

Graduate College Representative, Dr. Emma Regentova, Ph.D